LAPORAN PRAKTIKUM DATA STRUCTURE

BINARY TREES

Dosen Pengampu:

H. Fatchurrochman,M.Kom

Asisten Praktikum:

Fillah Anjany 230605110033

Oleh :

Muhammad Alif Mujaddid **240605110082**

Jurusan Teknik Informatika Fakultas Sains dan Teknologi

UIN Maulana Malik Ibrahim Malang

2025

Kesimpulan yang diperoleh dari pembahasan praktikum kali ini adalah:

1. Tentang proses insert pada binary tree

Proses *insert* pada Binary Tree dilakukan dengan menempatkan node baru sesuai dengan aturan struktur Binary Search Tree (BST), yaitu:

- Jika nilai data baru lebih kecil dari node saat ini, maka node baru akan ditempatkan di subtree kiri.
- Jika nilai data baru lebih besar, maka akan ditempatkan di subtree kanan. Proses ini dilakukan secara rekursif atau berulang hingga ditemukan posisi kosong di kiri atau kanan tempat node baru dapat disisipkan. Dengan cara ini, tree tetap teratur sehingga pencarian data di kemudian hari menjadi lebih efisien.

2. Tentang proses find pada binary tree

Proses *find* pada Binary Tree dilakukan dengan membandingkan nilai yang dicari dengan nilai pada setiap node secara berurutan mengikuti aturan Binary Search Tree (BST). Jika nilai yang dicari lebih kecil dari node saat ini, maka pencarian dilanjutkan ke subtree kiri; sebaliknya, jika lebih besar, pencarian berpindah ke subtree kanan. Proses ini berlanjut hingga ditemukan node dengan nilai yang sesuai atau hingga mencapai node kosong (null) yang menandakan bahwa data tidak ditemukan.
Dengan cara ini, proses pencarian menjadi lebih cepat dan efisien karena tidak perlu menelusuri seluruh node, melainkan hanya mengikuti jalur tertentu berdasarkan perbandingan nilai.

3. Tentang proses delete pada binary tree

Proses *delete* pada Binary Tree bertujuan untuk menghapus suatu node dari tree dengan tetap mempertahankan struktur Binary Search Tree (BST). Dalam implementasinya, terdapat tiga kondisi utama yang harus diperhatikan:

a. Node yang dihapus tidak memiliki anak (leaf node): node tersebut langsung dihapus dari tree.
b. Node yang dihapus memiliki satu anak: node tersebut digantikan oleh anaknya agar hubungan antar-node tetap terjaga.
c. Node yang dihapus memiliki dua anak: node tersebut digantikan oleh successor, yaitu node dengan nilai terkecil pada subtree kanannya.
Dengan memperhatikan ketiga kondisi tersebut, tree tetap terstruktur dengan benar sehingga operasi pencarian dan penelusuran berikutnya tetap berjalan efisien.

4. Tentang operasi traverse

   Operasi *traverse* pada Binary Tree adalah proses menelusuri seluruh node dalam tree untuk menampilkan atau memproses setiap data yang ada. Dalam praktikum ini digunakan tiga jenis traversal utama, yaitu:

   a. **Preorder (Root–Left–Right):** menampilkan data dimulai dari root, kemudian anak kiri, dan terakhir anak kanan.
   b. **Inorder (Left–Root–Right):** menelusuri subtree kiri terlebih dahulu, kemudian menampilkan root, dan terakhir subtree kanan. Traversal ini menghasilkan data dalam urutan yang terurut (ascending) pada Binary Search Tree.
   c. **Postorder (Left–Right–Root):** menelusuri anak kiri dan kanan terlebih dahulu, lalu menampilkan root di akhir.
      Melalui operasi *traverse*, kita dapat memahami struktur dan isi tree secara keseluruhan sesuai dengan urutan penelusuran yang diinginkan.